2021/12/2　木　第 2 時限

内容（講義と演習）

1．MySQL Trigger (トリガー)の作成(11/25　講義)

2．wineShopAplus へのアクセス用トリガーの作成（12/2 作成演習）

MySQL ver8 以降，Grant Create Trigger Privilege に
「log_bin_trust_function_creators=1」の設定が必要です.

wineShopAplus のスキーマ

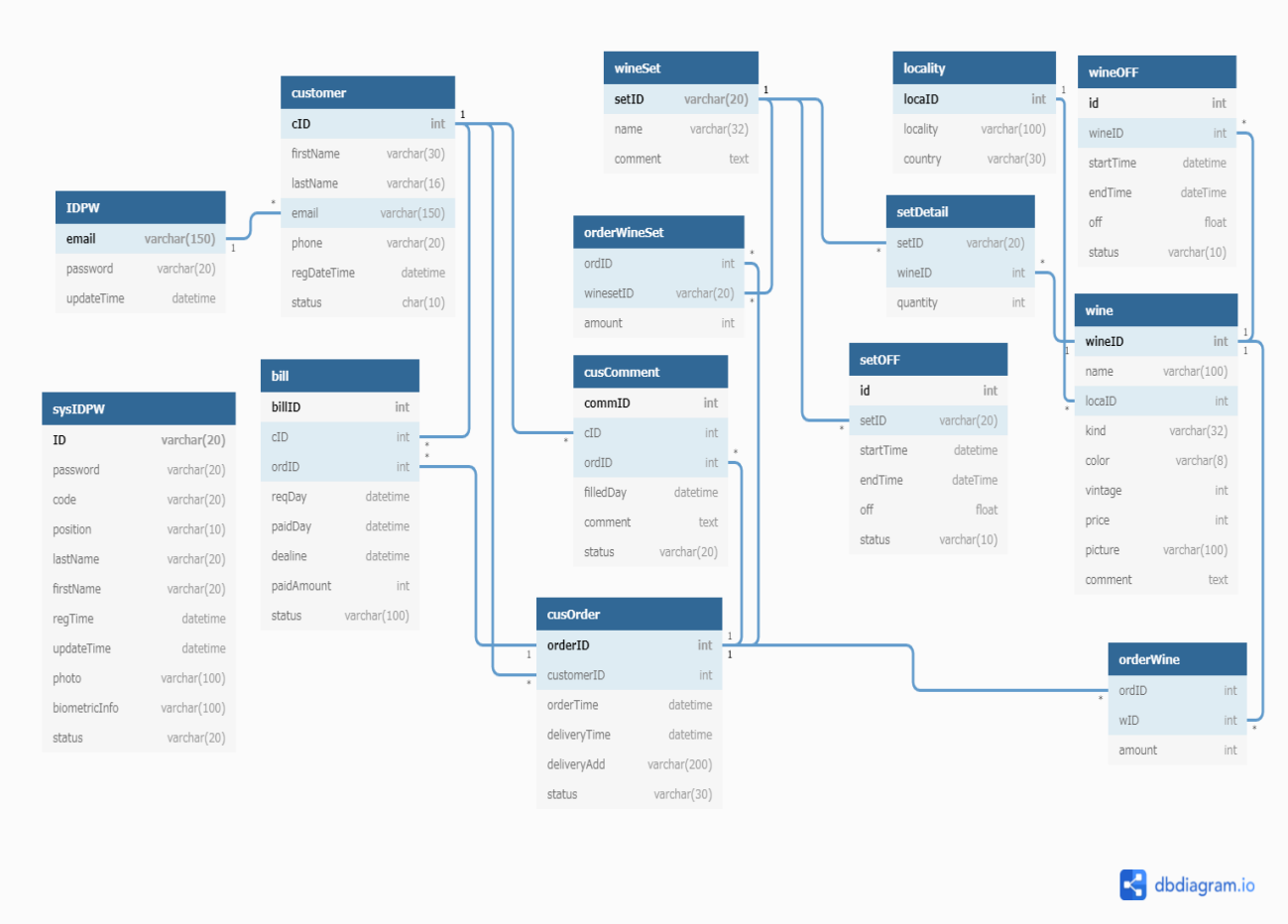

図 1.　wineShopAplus の ERD

MySQL Trigger (トリガー)とその作成例

About trigger:

トリガーはＳＱＬの insert, update, delete などテーブルを変える操作によるテーブルの変化をログファイルに保存したり，さらにこれらの変更を利用したりに有効である.

A. insert 操作によるトリガー

例題１テーブル IDPW の挿入操作を監察する用トリガーを作成し，その働きを検証する.

① まず，IDPW の構成と現存データを確認する.

```
mysql> desc IDPW;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| email      | varchar(150) | NO   | PRI | NULL    |       |
| password   | varchar(20)  | YES  |     | NULL    |       |
| updateTime | datetime     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> select * from IDPW;
+-----------------+----------+---------------------+
| email           | password | updateTime          |
+-----------------+----------+---------------------+
| Alpha@where.com | Here1    | 2020-06-30 12:47:29 |
| Bega@where.com  | Here11   | 2020-06-30 12:48:05 |
| Gama@where.com  | Here22   | 2020-06-30 12:48:37 |
+-----------------+----------+---------------------+
3 rows in set (0.00 sec)
```

② idpw 挿入監察用のログテーブル（log table，ログファイルともいう）を作る.

```
create table IDPWlog(
  email varchar(150),
  password  varchar(20),
  updateTime datetime
```

```
);
```

③ トリガーを作成する.

```
delimiter //
create trigger log_IDPW
 after insert on IDPW
 for each row
 begin
  insert into IDPWlog values(new.email, new.password, now());
 end;
//
delimiter ;
```

実装例

ログテーブルの作成

```
mysql> create table IDPWlog(
-> email varchar(150),
-> password    varchar(20),
-> updateTime datetime
-> );
Query OK, 0 rows affected (0.00 sec)
```

トリガーの作成

```
mysql> delimiter //
mysql> create trigger log_IDPW          トリガーを作成する
-> after insert on IDPW                 テーブル IDPW に追加した後
-> for each row                         行毎に
-> begin                                ログテーブル IDPWlog に記録する
-> insert into IDPWlog values(new.email, new.password, now());
->    end;
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

検証例

① IDPW に新しいデータを追加する.

```
mysql> insert into IDPW values('akira@where.com','AKIRA2020',now());
Query OK, 1 row affected (0.00 sec)
```

追加の確認

```
mysql> select * from IDPW;
+----------------+----------+---------------------+
| email          | password | updateTime          |
+----------------+----------+---------------------+
| akira@where.com | AKIRA2020 | 2020-09-25 12:18:23 |
| Alpha@where.com | Here1     | 2020-06-30 12:47:29 |
| Bega@where.com  | Here11    | 2020-06-30 12:48:05 |
| Gama@where.com  | Here22    | 2020-06-30 12:48:37 |
+----------------+----------+---------------------+
4 rows in set (0.00 sec)
```

② ログファイル（テーブル）IDPWlog を確認します.

```
mysql> select * from IDPWlog;
+----------------+----------+---------------------+
| email          | password | updateTime          |
+----------------+----------+---------------------+
| akira@where.com | AKIRA2020 | 2020-09-25 12:18:23 |
+----------------+----------+---------------------+
1 row in set (0.00 sec)
```

ログを取りました.

例題２. テーブル customer に新しい顧客を追加する。それをログテーブル（ファイル）に記録する.

① まず，customer の構成と現存データを確認する.

```
mysql> desc customer;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| cID        | int(11)     | NO   | PRI | NULL    | auto_increment |
| firstName  | varchar(30) | YES  |     | NULL    |                |
| lastName   | varchar(16) | YES  |     | NULL    |                |
| email      | varchar(150)| YES  | MUL | NULL    |                |
| phone      | varchar(20) | YES  |     | NULL    |                |
| regDateTime| datetime    | YES  |     | NULL    |                |
```

```
| status      | char(10)     | YES  |     | NULL    |               |
+-------------+--------------+------+-----+---------+---------------+
7 rows in set (0.00 sec)
mysql> select * from  customer;
+-----+-----------+-----------+----------------+-------+---------------------+---------+
| cID | firstName | lastName  | email          | phone | regDateTime         | status  |
+-----+-----------+-----------+----------------+-------+---------------------+---------+
|   1 | 桜        | 東山      | Alpha@where.com | NULL  | 2020-06-30 12:52:12 | offline |
|   2 | 健        | 山村      | Bega@where.com  | NULL  | 2020-06-30 12:53:07 | offline |
|   3 | 紫        | 伊集院    | Gama@where.com  | NULL  | 2020-06-30 12:54:05 | offline |
+-----+-----------+-----------+----------------+-------+---------------------+---------+
3  rows in set (0.00 sec)
```

② ログテーブルを作成しておく.

```
create table customerLog(
      cID   int(11),
   firstName    varchar(30),
   lastName    varchar(16),
   email         varchar(150),
   phone        varchar(20),
   regDateTime   datetime,
   status    char(10));
```

③ トリガーを作成する.

```
delimiter //
create  trigger  log_customer        トリガーを作成する
    after  insert  on  customer              テーブル customer に追加した後
    for each row                          行毎に
    begin                                 ログテーブル userlog に記録する
    insert  into  customerLog  values(new.cID, new. firstName,
        new.lastName,  new.email, new.phone, new.regDateTime,
        new.status);
    end;
 //
delimiter ;
```

変数 new の利用に注意：insert, update は new を利用可能，delete には使えない.

実装例

ログテーブルの作成

```
mysql>     create table customerLog(
    -> cID    int(11),
    -> firstName     varchar(30),
    -> lastName     varchar(16),
    ->    email          varchar(150),
    -> phone          varchar(20),
    -> regDateTime    datetime,
    -> status     char(10));

Query OK, 0 rows affected (0.00 sec)
```

トリガーの作成

```
mysql> delimiter //
mysql> create    trigger    log_customer
    -> after    insert    on    customer
    -> for each row
    -> begin
    -> insert into customerLog values(new.cID, new.firstName,
new.lastName, new.email, new.phone, new.regDateTime, new.status);
    -> end;
    -> //
mysql> delimiter ;
```

検証例

① 新しい顧客を追加する.

```
mysql> insert into customer(firstName, lastName, email, phone, regDateTime, status) values('
彬','浅倉',' akira@where.com ','090-1234-5678',now(),'active');
Query OK, 1 row affected (0.00 sec)
```

追加の確認

```
mysql> select * from customer;
+-----+-----------+----------+----------------+--------------+---------------------+---------+
| cID | firstName | lastName | email          | phone        | regDateTime         | status  |
+-----+-----------+----------+----------------+--------------+---------------------+---------+
|   1 | 桜        | 東山     | Alpha@where.com | NULL         | 2020-06-30 12:52:12 | offline |
|   2 | 健        | 山村     | Bega@where.com  | NULL         | 2020-06-30 12:53:07 | offline |
|   3 | 紫        | 伊集院   | Gama@where.com  | NULL         | 2020-06-30 12:54:05 | offline |
|   5 | 彬        | 浅倉     | akira@where.com | 090-1234-5678 | 2020-09-25 12:20:53 | active  |
```

```
+-----+----------+----------+---------------+--------------+--------------------+---------+
4 rows in set (0.00 sec)
```

② ログテーブル（ファイル）を確認する.

```
mysql> select * from customerLog;
+------+----------+----------+---------------+--------------+--------------------+--------+
| cID  | firstName | lastName | email         | phone        | regDateTime        | status |
+------+----------+----------+---------------+--------------+--------------------+--------+
|    5 | 彬        | 浅倉     | akira@where.com | 090-1234-5678 | 2020-09-25 12:30:53 | active |
+------+----------+----------+---------------+--------------+--------------------+--------+

1 row in set (0.00 sec)
ログを取りました.
```

注意：ログテーブルは普通に update 命令を使いデータを変えるので，データ更新できないようログテーブルに権限を設定すること.

```
          |
+--------+-----------+--------------------+
|        1 | abe@japan.com | 2017-07-19 11:19:03 |
+--------+-----------+--------------------+
1 row in set (0.00 sec)

ログを取りました.
```

B. update によるトリガー

例題３. テーブル wineSet を update する前にデータのログを取る。

まず，テーブル wineSet の構成と現存のデータを確認する.

```
mysql> desc wineSet;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| setID | varchar(20) | NO   | PRI | NULL    |       |
| name  | varchar(32) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> select * from wineSet;
+-------+------------------------------+
| setID | name                         |
+-------+------------------------------+
| KW-1  | 甲州白ワインセット           |
| KW-2  | 甲州紅白ワインセット         |
| s-1   | ブルゴーニュセット           |
| s-2   | ボルドーセット               |
| s-3   | 白ワインセット               |
| s-4   | 赤ワインセット               |
+-------+------------------------------+
5     rows in set (0.00 sec)
```

次に，ログテーブルを作成しておく．

```
create table wineSetLog(setID    varchar(20), oldName varchar(32));
```

最後に，トリガーを作成する．更新する前のデータを取っておく．

```
create trigger log_wineSet
    before update on wineSet
    for each row
    begin
        insert into wineSetLog(setID, oldName)
        values(old.setID,old.name);
    end;
```

実装例

```
mysql> create table wineSetLog(setID    varchar(20), oldName varchar(32));
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter //
mysql> create trigger log_wineSet
    -> before update on wineSet
    -> for each row
    -> begin
    -> insert into wineSetLog(setID, oldName) values(old.setID,old.name);
```

```
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

検証例

KW-1 の甲州白ワインセットの名前を'甲州白セット'に改名する.

```
mysql> update wineSet set name='甲州白セット' where setID="KW-1";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

更新結果の確認：

```
mysql> select * from wineSet where setID='KW-1';
+-------+--------------------+
| setID | name               |
+-------+--------------------+
| KW-1  | 甲州白セット       |
+-------+--------------------+
1 row in set (0.00 sec)
```

ログの確認：

```
mysql> select * from wineSetLog;
+-------+---------------------------+
| setID | oldName                   |
+-------+---------------------------+
| KW-1  | 甲州白ワインセット        |
+-------+---------------------------+
1 row in set (0.00 sec)
```

更新する前の名前のログを取りました.

変数 old に注意：delete, update は old を利用可能、insert には使えない。

例題 4 テーブル wine の価格(price)の変更によるトリガーを作成せよ．ログファイルは，wPriceLog(wineID，元の価格，変更日付)　のように作る.

① ログファイルを作る：

```
create table wPriceLog(
wineID int,
LastPrice int,
updateDT datetime);
```

②trigger log_wPrice を作る

```
delimiter //
create trigger log_wPrice
 before update on wine
 for each row
 begin
  insert into wPriceLog values(old.wineID,old.price,now());
 end;
//
 delimiter ;
```

実装例

```
mysql> create table wPriceLog(
    -> wineID int,
    -> LastPrice int,
    -> updateDT datetime);
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter //
mysql> create trigger log_wPrice
    ->   before update on wine
    ->   for each row
    ->   begin
    ->    insert into wPriceLog values(old.wineID,old.price,now());
    ->   end;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>   delimiter ;
```

② 検証： まず，更新する前の値段を見る．

```
mysql> select wineID,name,color,price from wine where wineID=11;
```

```
+--------+--------+-------+-------+
| wineID | name   | color | price |
+--------+--------+-------+-------+
|     11 | 雪花   | 赤    |   886 |
+--------+--------+-------+-------+
1 row in set (0.00 sec)
```

次に，値段を更新する．

```
mysql> update wine set price=price*1.1 where wineID=11;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

更新後のデータを確認する．

```
mysql> select wineID, name, color, price from wine where wineID=11;
+--------+--------+-------+-------+
| wineID | name   | color | price |
+--------+--------+-------+-------+
|     11 | 雪花   | 赤    |   975 |
+--------+--------+-------+-------+
1 row in set (0.00 sec)
```

ログファイルを確認する．

```
mysql> select * from wPriceLog;
+--------+-----------+---------------------+
| wineID | LastPrice | updateDT            |
+--------+-----------+---------------------+
|     11 |       886 | 2020-09-25 13:13:05 |
+--------+-----------+---------------------+
1 row in set (0.00 sec)
```

更新する前の値段のログを取った．

C. About the Types of triggers

トリガーのタイプは after トリガーと before トリガー2 種類がある。

after トリガーはテーブルに追加されたと更新された行を変えない。

before トリガーはテーブルの対応した行を変えられる。

例題５．ワインの価格入力する際，負数に入力したとき TRIGGER で 100 に書き換える．
まず，ログファイルを作成しておく．

```
create table logWine( name varchar(20));
```

トリガーを作成する。

```
create trigger wineT
before insert on wine
for each row
begin
   if new.price <=0 then
      set new.price=100;
      insert into logWine values(new.name);
      end if;
end
```

実行例：

```
insert into wine(name,price) values("test",-10);
insert into wine(name,price) values(“correct”,1000);
```

```
mysql> select name,price from wine where name="test";
+------+-------+
| name | price |
+------+-------+
| test |   100 |
+------+-------+
1 row in set (0.00 sec)

mysql> select name,price from wine where name="correct";
+---------+-------+
| name    | price |
+---------+-------+
| correct |  1000 |
+---------+-------+
1 row in set (0.00 sec)
```

ログファイル：

```
mysql> select * from logWine;
+------+
| name |
+------+
| test |
+------+
1 row in set (0.00 sec)
```

例題６．wine の price の更新入力における入力ミスを防ぐことができるトリガーを作る．

① ログファイルの作成

スキーマ:

winePriceLog(wineID int,name varchar(100),lastPrice int, updateDate datetime, error int)

② トリガーを作成

入力ミスがない場合：error を 0 にします．更新する前の値段を記録します．

ミスがある場合：error を 1 にします．値段更新せずに間違った入力を記録します．

入力ミスの定義：正しいデータは 700～１０万円とし，それ以外はエラーとします．

正しいの場合（条件）

if new.price between 700 and 100000 または

if new.price>= 700 and new.price <= 100000

```
delimiter //
create trigger update_winePrice
before update on wine
for each row
begin
if new.price between 700 and 100000   then
  insert into winePriceLog values(old.wineID, old.name,old.price,now(),0);
else
  insert into winePriceLog values(old.wineID, old.name,new.price,now(),1);
  set new.price = old.price;
end if;
end;
//
delimiter ;
```

検証：

例１：正しい入力をした場合：

```
mysql> select    wineID, price from wine;
+--------+-------+
| wineID | price |
+--------+-------+
|      1 |  2400 |
|      2 |  3000 |
|      3 |  5800 |
|      4 |  2200 |
|      5 |  3080 |
|      6 |  4000 |
|      7 |  2580 |
|      8 |  1859 |
|      9 |  1901 |
|     10 |  2484 |
|     11 |  2400 |
|     12 |  2000 |
|     13 |  1000 |
+--------+-------+
13 rows in set (0.00 sec)

mysql> select * from winePrice_log;
+--------+-------------+-----------+--------------------+------+
| wineID | name        | lastPrice | updateDate         | erro |
+--------+-------------+-----------+--------------------+------+
|      1 | シャブリ    |        10 | 2017-07-26 11:23:09 |    1 |
|      1 | シャブリ    |      2640 | 2017-07-26 11:24:30 |    0 |
|      1 | シャブリ    |      3500 | 2017-07-26 11:26:39 |    0 |
|      1 | シャブリ    |        24 | 2017-07-26 11:27:57 |    1 |
|      1 | シャブリ    |      2400 | 2017-07-26 11:59:52 |    0 |
+--------+-------------+-----------+--------------------+------+
5 rows in set (0.00 sec)

mysql> update wine set price=2500 where wineID=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from winePrice_log;
+--------+-------------+-----------+--------------------+------+
| wineID | name        | lastPrice | updateDate         | erro |
+--------+-------------+-----------+--------------------+------+
|      1 | シャブリ    |        10 | 2017-07-26 11:23:09 |    1 |
|      1 | シャブリ    |      2640 | 2017-07-26 11:24:30 |    0 |
|      1 | シャブリ    |      3500 | 2017-07-26 11:26:39 |    0 |
|      1 | シャブリ    |        24 | 2017-07-26 11:27:57 |    1 |
|      1 | シャブリ    |      2400 | 2017-07-26 11:59:52 |    0 |
|      1 | シャブリ    |      2400 | 2017-07-26 12:38:23 |    0 |
+--------+-------------+-----------+--------------------+------+
6 rows in set (0.00 sec)

mysql> select    wineID, price from wine;
+--------+-------+
| wineID | price |
+--------+-------+
|      1 |  2500 |
|      2 |  3000 |
|      3 |  5800 |
|      4 |  2200 |
|      5 |  3080 |
|      6 |  4000 |
|      7 |  2580 |
|      8 |  1859 |
|      9 |  1901 |
|     10 |  2484 |
|     11 |  2400 |
|     12 |  2000 |
|     13 |  1000 |
+--------+-------+
13 rows in set (0.00 sec)
```

C. 間違った入力をした場合：

```
mysql> select    wineID, price from wine;
+--------+-------+
| wineID | price |
+--------+-------+
```

```
|         1 |   2500 |
|         2 |   3000 |
|         3 |   5800 |
|         4 |   2200 |
|         5 |   3080 |
|         6 |   4000 |
|         7 |   2580 |
|         8 |   1859 |
|         9 |   1901 |
|        10 |   2484 |
|        11 |   2400 |
|        12 |   2000 |
|        13 |   1000 |
+--------+-------+
13 rows in set (0.00 sec)

mysql> select * from winePrice_log;
+--------+-------------+----------+--------------------+------+
| wineID | name        | lastPrice | updateDate         | erro |
+--------+-------------+----------+--------------------+------+
|         1 | シャブリ       |           10 | 2017-07-26 11:23:09 |      1 |
|         1 | シャブリ       |         2640 | 2017-07-26 11:24:30 |      0 |
|         1 | シャブリ       |         3500 | 2017-07-26 11:26:39 |      0 |
|         1 | シャブリ       |           24 | 2017-07-26 11:27:57 |      1 |
|         1 | シャブリ       |         2400 | 2017-07-26 11:59:52 |      0 |
|         1 | シャブリ       |         2400 | 2017-07-26 12:38:23 |      0 |
+--------+-------------+----------+--------------------+------+
6 rows in set (0.00 sec)

mysql> update wine set price=25 where wineID=1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select  wineID, price from wine;
+--------+-------+
| wineID | price |
+--------+-------+
|         1 |   2500 |
```

```
|        2 |   3000 |
|        3 |   5800 |
|        4 |   2200 |
|        5 |   3080 |
|        6 |   4000 |
|        7 |   2580 |
|        8 |   1859 |
|        9 |   1901 |
|       10 |   2484 |
|       11 |   2400 |
|       12 |   2000 |
|       13 |   1000 |
+--------+-------+
13 rows in set (0.00 sec)

mysql> select * from winePrice_log;
+--------+-------------+-----------+--------------------+------+
| wineID | name        | lastPrice | updateDate         | erro |
+--------+-------------+-----------+--------------------+------+
|      1 | シャブリ    |        10 | 2017-07-26 11:23:09 |    1 |
|      1 | シャブリ    |      2640 | 2017-07-26 11:24:30 |    0 |
|      1 | シャブリ    |      3500 | 2017-07-26 11:26:39 |    0 |
|      1 | シャブリ    |        24 | 2017-07-26 11:27:57 |    1 |
|      1 | シャブリ    |      2400 | 2017-07-26 11:59:52 |    0 |
|      1 | シャブリ    |      2400 | 2017-07-26 12:38:23 |    0 |
|      1 | シャブリ    |        25 | 2017-07-26 12:40:12 |    1 |
+--------+-------------+-----------+--------------------+------+
7 rows in set (0.00 sec)
```

D. トリガー作成における注意点

例題 7. 正しいではないトリガーの例。

要求： 新しいユーザーを追加する際、パスワードを未指定の場合は”newTest”をパスワードとして登録する。そのユーザーの名前をログする。

```
mysql> desc user;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| ID       | int(11)     | NO   | PRI | NULL    | auto_increment |
| firstName| varchar(30) | YES  |     | NULL    |                |
| lastName | varchar(30) | YES  |     | NULL    |                |
| sex      | char(6)     | YES  |     | NULL    |                |
| password | char(16)    | NO   |     | NULL    |                |
| regDay   | datetime    | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

① ログファイルを作っておく。

```
create table userLog(
        firstName char(20),
         lastName char(20)
    );
```

② トリガーを作る。

```
mysql> delimiter //
mysql> create trigger    userpass
    -> before insert on user
    -> for each row
    -> begin
    -> if new.password=null    then
    -> set new.password="newTest";
    -> insert into userlog values(new.firstName,new.lastName);
    -> end if;
    -> end
    -> //
```

新しいユーザーを登録する際、以下のような insert 命令を組めば、問題ない。

```
mysql> create trigger userpass before insert on user for each row begin  if new.pas
sword=null then set new.password="newpass"; insert into userLog values(new.firstNam
e,new.lastName); end if; end//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> insert into user(firstName,lastName,password) values('Red','Yama','qwer');
Query OK, 1 row affected (0.00 sec)

mysql> select * from user;
+-----+------------+----------+------+----------+---------------------+
| ID  | firstName  | lastName | sex  | password | regDay              |
+-----+------------+----------+------+----------+---------------------+
| 300 | Jone       | Smith    | F    | who      | NULL                |
| 302 | Higashiyama | Akiko   | F    | akiko    | 2015-09-23 12:08:19 |
| 305 | Red        | Yama     | NULL | qwer     | NULL                |
+-----+------------+----------+------+----------+---------------------+
3 rows in set (0.00 sec)

mysql> insert into user(firstName,lastName) values('Blue','SEA');
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> select * from user;
+-----+------------+----------+------+----------+---------------------+
| ID  | firstName  | lastName | sex  | password | regDay              |
+-----+------------+----------+------+----------+---------------------+
| 300 | Jone       | Smith    | F    | who      | NULL                |
| 302 | Higashiyama | Akiko   | F    | akiko    | 2015-09-23 12:08:19 |
| 305 | Red        | Yama     | NULL | qwer     | NULL                |
| 306 | Blue       | SEA      | NULL |          | NULL                |
+-----+------------+----------+------+----------+---------------------+
4 rows in set (0.00 sec)
```

ログファイルの中身は空である。

```
mysql> select * from userLog;
Empty set (0.00 sec)
```

しかし、次の命令は実行されない。トリガーも無効である。原因はテーブル user の定義で password は NOT NULL に指定されたためである。

```
mysql> insert into user(firstName,lastName,password) values('Yellow','Flower',null);

ERROR 1048 (23000): Column 'password' cannot be null
```

修正案：password に dummy string を使う。

create trigger userpass before insert on user for each row begin if new.password="test" then set new.password="newpass"; insert into userLog values(new.firstName,new.lastName); end if; end

```
mysql> insert into user(firstName,lastName,password) values("Green","Beans","test");

Query OK, 1 row affected (0.01 sec)

mysql> select * from user;
+-----+-------------+----------+------+----------+---------------------+
| ID  | firstName   | lastName | sex  | password | regDay              |
+-----+-------------+----------+------+----------+---------------------+
| 300 | Jone        | Smith    | F    | who      | NULL                |
| 302 | Higashiyama | Akiko    | F    | akiko    | 2015-09-23 12:08:19 |
| 305 | Red         | Yama     | NULL | qwer     | NULL                |
| 306 | Blue        | SEA      | NULL |          | NULL                |
| 307 | Green       | Beans    | NULL | newpass  | NULL                |
+-----+-------------+----------+------+----------+---------------------+
5 rows in set (0.00 sec)
```

ログファイルの中身：

```
mysql> select * from userLog;
+-----------+----------+
| firstName | lastName |
+-----------+----------+
| Green     | Beans    |
+-----------+----------+
1 row in set (0.00 sec)
```

E. トリガーの確認

show triggers;

F. トリガーの廃棄

DROP TRIGGER TRIGGER_NAME;

演習課題 dbxx_1202：wineShopAplus へのアクセス用トリガーの作成

例題 1,2, 4 を実装してください．課題ごとに追加や更新操作およびそれらに対応しているトリガーの実行結果を収めたテキストファイル dbxx_1202.txt を sningping@kumamoto-nct.ac.j へ添付で提出ください.